IJAMML 3:2 (2015) 137-162 December 2015
ISSN: 2394-2258
Available at http://scientificadvances.co.in
DOI: http://dx.doi.org/10.18642/ijamml_7100121564

IMPROVED LEAST SQUARES TWIN SUPPORT MATRIX MACHINES

Xizhan Gao[a], Liya Fan[b] and Haitao Xu[b]

[a]School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, Jingsu, P. R. China

[b]School of Mathematical Sciences, Liaocheng University, Liaocheng, Shandong, P. R. China

Abstract

In this paper, a new classification method named as improved least squares twin support matrix machines (ILS-TSMM) is presented, which is an improvement of linear support tensor machines (STM) and linear least squares twin support vector machines (LS-TSVM). We know that nonlinear classifiers for vector data are a lot of but for matrix data are few. In order to study the nonlinear version of ILS-TSMM (NILS-TSMM), a new matrix kernel function is introduced and based on which, we provide a detailed theoretical derivation for NILS-TSMM. The advantage of our methods is that the structural risk minimization problem is considered by introducing the regularization term and that the utility of twin skill and least squares technology aims to reduce the computation time (training time and testing time). The experiment results show that the proposed methods are effective and efficient.

Keywords: twin support matrix machine, least squares technology, matrix data classification, matrix kernel function, iterative algorithm.

*Corresponding author.
E-mail address: gaoxizhan123@126.com (Xizhan Gao).

2010 Mathematics Subject Classification: 68T10.
Submitted by Jianqiang Gao.
Received October 29, 2015

1. Introduction

In the past decades, numerous classification methods have been proposed and successfully applied to many fields. All these methods can be divided into two types, vector classifiers and tensor classifiers. The vector-based classifiers are oriented to vector, such as support vector machine (SVM) [1, 2], twin support vector machine (TSVM) [3], improved twin support machine (ITSVM) [4], least squares support vector machine (LS-SVM) [5, 6], least squares twin support vector machine (LS-TSVM) [7] and so on. Some extensions for TSVM can be found in [8-18]. SVM was proposed by Vapnik for binary classification, which aims to find a pair of parallel boundary hyper-planes with maximum margin. Different from SVM, TSVM aims to find a pair of nonparallel hyper-planes by solving two smaller quadratic programming problems (QPPs) such that each plane is closer to one of two classes and at least one distance from the other. TSVM has the similar formulation with SVM except that not all the patterns appear in the constraints of either problem at the same time, which makes the learning speed of TSVM is more faster than that of SVM. LS-TSVM is a least squares version of TSVM which uses the equality constraints instead of inequality constraints and the square of 2-norm of slack variables with weight $\frac{c}{2}$ instead of 1-norm with weight c. This leads to a simple and fast algorithm.

Matrices are common forms of data that are encountered in a wide range of real applications, such as photorealistic images of faces, palms, and medical images. How to classify this kind of data is an important research topic for pattern recognition and machine learning [19, 20]. Although vector-based classifiers and their variants have achieved satisfactory performance in many cases, they may lack efficiency in managing matrix data by simply reformulating them into vectors. The main reasons are as follows: (1) When we reformulate a matrix as a vector, the dimensionality of this vector is often very high, which may leads to the cause of dimensionality problem and the small sample size problem (the dimension of input data is much higher than the number of

input data). (2) With the increase of dimensionality, the computation time will increase drastically. (3) When a matrix is collapsed as a vector, the spatial correlations of the matrix will be lost. In order to solve these problems, in recent years, a lot of researchers have been conducted on tensor-based approaches for image data analysis and some linear classifiers have been developed for pattern classification, for details see [21-31]. However, up to now, nonlinear classifiers for matrix data are few.

Motivated by the above works, in this paper, we will study the matrix version of LS-TSVM and simultaneously consider the structural risk minimization problem for matrix data classification. We will propose a new classifier named as improved least squares twin support matrix machines (ILS-TSMM), which is an extension of least squares twin support tensor machines (LS-TSTM) and least squares twin support vector machines (LSTSVM). In order to study the nonlinear version of ILS-TSMM (NILS-TSMM), a new matrix kernel function is introduced and based on which, we provide a detailed theoretical derivation for NILS-TSMM. The advantage of our methods is that the structural risk minimization principle is implemented by introducing the regularization term and that the utility of twin skill and least squares technology aims to reduce the computation time (training time and testing time).

The rest of the paper is organized as follows. In Section 2, background and related works are introduced. In Section 3, we first present ILS-TSMM in an iterative framework and then provide a detailed theoretical derivation for NILS-TSMM by means of a new defined matrix kernel function. Experiments and results analysis are performed in Section 4. Section 5 gives some conclusions.

2. Background and Related Works

2.1. Linear twin support vector machines (linear TSVM)

In this subsection, we will recall briefly linear TSVM (for details, see [3]). The basic idea of linear TSVM is constructing a pair of nonparallel hyper-planes such that each one is close as possible as to one class, and

far as possible as from the other class. A new input will be assigned to one of the classes depending on its proximity to which hyper-plane. Let $T = \{(x_i, y_i)\}_{i=1}^{m}$ be a set of vector sample data, where $x_i \in R^n$ and $y_i \in \{\pm 1\}$ are the input and the class label of the i-th sample data, respectively. Let m_1 and m_2 be, respectively, the numbers of positive and negative sample data and $m = m_1 + m_2$. We denote by $A \in R^{n \times m_1}$ the matrix composed of positive samples and by $B \in R^{n \times m_2}$ the matrix composed of negative samples. Linear TSVM seeks a pair of nonparallel hyper-planes $f_1(x) = w_1^T x + b_1 = 0$ and $f_2(x) = w_2^T x + b_2 = 0$ by considering the following two quadratic programming problems (QPPs):

$$\min_{w_1, b_1, \xi_2} \frac{1}{2}\|Aw_1 + e_1 b_1\|^2 + c_1 e_2^T \xi_2$$

$$s.t. -(B^T w_1 + e_2 b_1) \geq e_2 - \xi_2, \quad \xi_2 \geq 0, \tag{1}$$

$$\min_{w_2, b_2, \xi_1} \frac{1}{2}\|Bw_2 + e_2 b_2\|^2 + c_2 e_1^T \xi_1$$

$$s.t. \ (A^T w_2 + e_1 b_2) \geq e_1 - \xi_1, \quad \xi_1 \geq 0, \tag{2}$$

where $w_1, w_2 \in R^n$ and $b_1, b_2 \in R$ are decision variables, $c_1, c_2 > 0$ are tradeoff parameters, $\xi_1 \in R^{m_1}$, $\xi_2 \in R^{m_2}$ are slack variables vectors, and $e_1 \in R^{m_1}$, $e_2 \in R^{m_2}$ are vectors of ones. By solving respectively the Wolfe dual forms of the problems (1) and (2), we can obtain (w_1, b_1) and (w_2, b_2). A new input $\tilde{x}$ is assigned the class $k (k = 1, 2)$ depending on which of the two hyper-planes is closer to, that is, $k = \arg\min\{\frac{|f_1(\tilde{x})|}{\|w_1\|}, \frac{|f_2(\tilde{x})|}{\|w_2\|}\}$.

2.2. Linear least squares twin support vector machines (linear LS-TSVM)

Linear LS-TSVM is an improvement of linear TSVM which uses the equality constraints instead of inequality constraints and the square of 2-norm of slack variables with weight $\frac{c}{2}$ instead of 1-norm with weight c (for details, see [7]). The primal problems of linear LS-TSVM are as follows:

$$\min_{w_1, b_1, \xi_2} \frac{1}{2}\|Aw_1 + e_1 b_1\|^2 + \frac{c_1}{2}\xi_2^T \xi_2$$

$$s.t. -(B^T w_1 + e_2 b_1) = e_2 - \xi_2, \quad \xi_2 \geq 0, \tag{3}$$

$$\min_{w_2, b_2, \xi_1} \frac{1}{2}\|Bw_2 + e_2 b_2\|^2 + \frac{c_2}{2}\xi_1^T \xi_1$$

$$s.t. \ (A^T w_2 + e_1 b_2) = e_1 - \xi_1, \quad \xi_1 \geq 0. \tag{4}$$

By solving directly the problems (3) and (4), not their Wolfe dual forms, we can obtain (w_1, b_1) and (w_2, b_2).

2.3. Linear support tensor machines (linear STM)

In this subsection, we recall briefly linear support tensor machines as a tensor generalization of SVM in the tensor space, for details see [21, 22]. Different from SVM, the main idea of STM is to deal with tensor inputs directly without vectorization, which keeps the structure information of image data and has not increase time complexity. Let $T = \{(X_i, y_i)\}_{i=1}^m$ be a set of second tensor samples, where $X_i \in R^{n_1} \otimes R^{n_2}$ and $y_i \in \{\pm 1\}$ are the input and the class label of the i-th second tensor sample, respectively. Linear STM finds a tensor classifier $f(X) = u^T X v + b$, where $u \in R^{n_1}$, $v \in R^{n_2}$ and $b \in R$, such that the two classes can be separated with maximum margin. The implement of STM is actually based on an iteration procedure, which is stated as follows.

Algorithm 1: Linear STM

1. Initialization. Let $t = 0$, $\varepsilon > 0$ be small enough and take $u^t = [1, \cdots, 1]^T \in R^{n_1}$.

2. Compute v^t. Let $x_i^t = X_i^T u^t$, $i = 1, \cdots, m$ and $\beta_1^t = \|u^t\|^2$. (v^t, b_1^t) can be computed by solving the following optimization problem with $\beta_1 = \beta_1^t$ and $x_i = x_i^t$:

$$\begin{aligned} &\min_{v, b_1, \xi} \frac{1}{2}\beta_1 v^T v + Ce^T \xi \\ &s.t.\ y_i(v^T x_i + b_1) + \xi_i \geq 1, \quad i = 1, \cdots, m, \\ &\xi_i \geq 0,\ i = 1, \cdots, m, \end{aligned} \tag{5}$$

where $C > 0$ is a parameter, $\xi \in R^m$ is a slack vector, and $e \in R^m$ is the vector of ones.

3. Update u^t. Let $\tilde{x}_i^t = X_i v^t$, $i = 1, \cdots, m$ and $\beta_2^t = \|v^t\|^2$. (u^{t+1}, b_2^{t+1}) can be computed by solving the following optimization problem with $\beta_2 = \beta_2^t$ and $\tilde{x}_i = \tilde{x}_i^t$:

$$\begin{aligned} &\min_{u, b_2, \xi} \frac{1}{2}\beta_2 u^T u + Ce^T \xi \\ &s.t.\ y_i(u^T \tilde{x}_i + b_2) + \xi_i \geq 1, \quad i = 1, \cdots, m, \\ &\xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned} \tag{6}$$

4. Update v^t. Let $x_i^{t+1} = X_i^T u^{t+1}$, $i = 1, \cdots, m$ and $\beta_1^{t+1} = \|u^{t+1}\|^2$. (v^{t+1}, b_1^{t+1}) can be computed by solving the problem (5) with $\beta_1 = \beta_1^{t+1}$ and $x_i = x_i^{t+1}$.

5. If $\|u^{t+1} - u^t\| < \varepsilon$, $\|v^{t+1} - v^t\| < \varepsilon$, or maximum number of iteration is achieved, put $u^* \leftarrow u^{t+1}$, $b_1^* \leftarrow b_1^{t+1}$, $v^* \leftarrow v^{t+1}$ and $b_2^* \leftarrow b_2^{t+1}$; otherwise, put $t \leftarrow t+1$ and return back Step 2.

6. Compute $b^* = \frac{1}{2}(b_1^* + b_2^*)$ and construct the decision function $f(X) = (u^*)^T X v^* + b^*$, $\forall X \in R^{n_1 \times n_2}$.

3. Improved Least Squares Twin Support Matrix Machines (ILS-TSMM)

We know that classifying matrix data by using vector-based classification algorithms must transform matrix samples into vectors firstly. When an $m \times n$ matrix data is scanned into a mn dimensional vector, its structure information will be lost. In order to protect the structure information of matrix data and at the same time reduce the calculation time, in this section, we study the matrix version of LS-TSVM and simultaneously consider the structural risk minimization problem. We first consider the linear case and propose a classification method, named as improved least squares twin support matrix machine (ILS-TSMM). Then, in order to study the nonlinear version of ILS-TSMM, a new matrix kernel function is introduced and based on which, we provide a detailed theoretical derivation for nonlinear version and suggest a nonlinear classification method, named as nonlinear ILS-TSMM (NILS-TSMM).

Let $T = \{(X_i, y_i)\}_{i=1}^m$ be a set of matrix samples, where $X_i \in R^{n_1 \times n_2}$ and $y_i \in \{\pm 1\}$ are the input and the class label of the i-th sample, respectively. Let m_1 and m_2 be respectively the numbers of positive and negative samples, I_1 and I_2 be respectively the index sets of positive and negative samples and $m = m_1 + m_2$. The norm of a matrix C is defined by $\|C\|^2 = Tr(C^T C)$, where $Tr(C)$ denotes the trace of the matrix C.

3.1. ILS-TSMM

ILS-TSMM aims to seek a pair of nonparallel hyper-planes $f_1(X) = u_1^T X v_1 + b_1 = 0$ and $f_2(X) = u_2^T X v_2 + b_2 = 0$ by considering the following two QPPs:

$$\min_{u_1, v_1, b_1, \xi} \frac{c_3}{2}(\|u_1 v_1^T\|^2 + b_1^2) + \frac{1}{2}\sum_{i \in I_1}(u_1^T X_i v_1 + b_1)^2 + \frac{c_1}{2}\sum_{j \in I_2}\xi_j^2$$

$$s.t. -(u_1^T X_j v_1 + b_1) + \xi_j = 1, \quad j \in I_2, \tag{7}$$

$$\min_{u_2, v_2, b_2, \eta} \frac{c_4}{2}(\|u_2 v_2^T\|^2 + b_2^2) + \frac{1}{2}\sum_{j \in I_2}(u_2^T X_j v_2 + b_2)^2 + \frac{c_2}{2}\sum_{i \in I_1}\eta_i^2$$

$$s.t. (u_2^T X_i v_2 + b_2) + \eta_i = 1, \quad i \in I_1, \tag{8}$$

where $u_1, u_2 \in R^{n_1}$, $v_1, v_2 \in R^{n_2}$, and $b_1, b_2 \in R$ are decision variables, $c_1, c_2, c_3, c_4 > 0$ are tradeoff parameters and $\xi \in R^{m_2}$, $\eta \in R^{m_1}$ are slack vectors. We implement the structural risk minimization by adding, respectively, the regularization terms $\frac{c_3}{2}(\|u_1 v_1^T\|^2 + b_1^2)$ and $\frac{c_4}{2}(\|u_2 v_2^T\|^2 + b_2^2)$ in objective functions of the problems (7) and (8). Once (u_1, v_1, b_1) and (u_2, v_2, b_2) are solved, a new input $\widetilde{X}$ is assigned the class $k(k = 1, 2)$ depending on which of the two hyper-planes is closer to, that is, $k = \arg\min\{\frac{|f_1(\widetilde{X})|}{\|u_1 v_1^T\|}, \frac{|f_2(\widetilde{X})|}{\|u_2 v_2^T\|}\}$.

Next, we solve (u_1, v_1, b_1) by an iterative framework. Similarly, we can solve (u_2, v_2, b_2).

We first consider the relationships among u_1, v_1 and b_1. Substituting the equality constraints into the objective function of the problem (7), we have

$$\min_{u_1, v_1, b_1, \xi} \frac{c_3}{2}(\|u_1 v_1^T\|^2 + b_1^2) + \frac{1}{2}\sum_{i \in I_1}(u_1^T X_i v_1 + b_1)^2 + \frac{c_1}{2}\sum_{j \in I_2}(1 + u_1^T X_j v_1 + b_1)^2. \tag{9}$$

Set the gradient of the objective function of the problem (9) with respect to (u_1, v_1, b_1) is zero, then we can deduce that

$$c_3 v_1^T v_1 u_1 + \sum_{i \in I_1} X_i v_1 v_1^T X_i^T u_1 + c_1 \sum_{j \in I_2} X_j v_1 v_1^T X_j^T u_1 + b_1 \sum_{i \in I_1} X_i v_1 + c_1 b_1 \sum_{j \in I_2} X_j v_1 + c_1 \sum_{j \in I_2} X_j v_1 = 0,$$

$$c_3 u_1^T u_1 v_1 + \sum_{i \in I_1} X_i^T u_1 u_1^T X_i v_1 + c_1 \sum_{j \in I_2} X_j^T u_1 u_1^T X_j v_1 + b_1 \sum_{i \in I_1} X_i^T u_1 + c_1 b_1 \sum_{j \in I_2} X_j^T u_1 + c_1 \sum_{j \in I_2} X_j^T u_1 = 0,$$

$$b_1 = -c_0\Big(\sum_{i \in I_1} u_1^T X_i v_1 + \sum_{j \in I_2} u_1^T X_j v_1\Big),$$

where $c_0 = \dfrac{1}{c_3 + m_1 + m_2 c_1}$, which indicates that u_1 and v_1 are dependent on each other and cannot be solved independently. Hence, we utilize an iteration method described as follows to solve (u_1, v_1, b_1).

On the one hand, for any given nonzero vector $v_1 \in R^{n_2}$, let $r_1 = \|v_1\|^2$, $x_i^T = X_i v_i$, $i = 1, \cdots, m$ and $A_1 \in R^{m_1 \times n_1}$ and $B_1 \in R^{m_2 \times n_1}$ be matrices composed of all x_i, $i \in I_1$ and all x_j, $j \in I_2$, respectively, then the problem (7) can be rewritten as

$$\min_{u_1, b_1, \xi} \frac{c_3}{2}(r_1\|u_1\|^2 + b_1^2) + \frac{1}{2}\|A_1 u_1 + e_1 b_1\|^2 + \frac{c_1}{2}\xi^T \xi$$

$$s.t. -(B_1 u_1 + e_2 b_1) + \xi = e_2, \tag{10}$$

where $e_1 \in R^{n_1}, e_2 \in R^{n_2}$ are vectors of ones. Substituting the equality constraints into the objective function of the problem (10) and setting the gradient of the objective function with respect to (u_1, b_1) to zero, we can deduce that

$$c_3 r_1 u_1 + A_1^T (A_1 u_1 + e_1 b_1) + c_1 B_1^T (B_1 u_1 + e_2 b_1 + e_2) = 0,$$

$$c_3 b_1 + e_1^T (A_1 u_1 + e_1 b_1) + c_1 e_2^T (B_1 u_1 + e_2 b_1 + e_2) = 0. \quad (11)$$

Let $E_1 = [A_1, e_1]$, $F_1 = [B_1, e_2]$, $\widetilde{E_1} = [\frac{1}{r_1} A_1, e_1]$, and $\widetilde{F_1} = [\frac{1}{r_1} B_1, e_2]$. We can obtain from (11) that

$$\begin{bmatrix} r_1 u_1 \\ b_1 \end{bmatrix} = -(\frac{c_3}{c_1} I + \frac{1}{c_1} E_1^T \widetilde{E_1} + F_1^T \widetilde{F_1})^{-1} F_1^T e_2. \quad (12)$$

On the another hand, for any given nonzero vector $u_1 \in R^{n_1}$, let $r_2 = \|u_1\|^2$ and $\tilde{x}_i^T = X_i^T u_1, i = 1, \cdots, m$ and $A_2 \in R^{m_1 \times n_2}$ and $B_2 \in R^{m_2 \times n_2}$ be matrices composed of all $\tilde{x}_i, i \in I_1$ and all $\tilde{x}_j, j \in I_2$, respectively, then the problem (7) can be rewritten as

$$\min_{v_1, d_1, \xi} \frac{c_3}{2} (r_2 \|v_1\|^2 + d_1^2) + \frac{1}{2} \|A_2 v_1 + e_1 d_1\|^2 + \frac{c_1}{2} \xi^T \xi$$

$$s.t. -(B_2 v_1 + e_2 d_1) + \xi = e_2. \quad (13)$$

By solving the problem (13), we can obtain

$$\begin{bmatrix} r_2 v_1 \\ d_1 \end{bmatrix} = -(\frac{c_3}{c_1} I + \frac{1}{c_1} E_2^T \widetilde{E_2} + F_2^T \widetilde{F_2})^{-1} F_2^T e_2, \quad (14)$$

where $E_2 = [A_2, e_1]$, $\widetilde{E_2} = [\frac{1}{r_2} A_2, e_1]$, $F_2 = [B_2, e_2]$, and $\widetilde{F_2} = [\frac{1}{r_2} B_2, e_2]$.

By the similar way, we can get that

$$\begin{bmatrix} \alpha_1 u_2 \\ b_2 \end{bmatrix} = (\frac{c_4}{c_2} I + \frac{1}{c_2} F_1^T \widehat{F_1} + E_1^T \widehat{E_1})^{-1} E_1^T e_1, \quad (15)$$

where $\alpha_1 = \|v_2\|^2$, $F_1 = [B_1,\ e_2]$, $\widehat{F_1} = [\ \frac{1}{\alpha_1} B_1,\ e_2\ ]$, $E_1 = [A_1,\ e_1]$, and $\widehat{E_1} = [\ \frac{1}{\alpha_1} A_1,\ e_1\ ]$.

$$\begin{bmatrix} \alpha_2 v_2 \\ d_2 \end{bmatrix} = (\ \frac{c_4}{c_2} I + \frac{1}{c_2} F_2^T \widehat{F_2} + E_2^T \widehat{E_2}\ )^{-1} E_2^T e_1, \tag{16}$$

where $\alpha_2 = \|u_2\|^2$, $F_2 = [B_2,\ e_2]$, $\widehat{F_2} = [\ \frac{1}{\alpha_2} B_2,\ e_2\ ]$, $E_2 = [A_2,\ e_1]$, and $\widehat{E_2} = [\ \frac{1}{\alpha_2} A_2,\ e_1\ ]$.

Summing up the discussion above, we can see that u_i and v_i $i = 1, 2$ can be obtained by iteratively solving (12) and (14), and(15)-(16). The specific procedure as follows.

Algorithm 2: ILS-TSMM

1. Initialization. Let $t = 0$, $\varepsilon > 0$ small enough and take $v_1^t = v_2^t = [1, \cdots, 1]^T \in R^{n_2}$.

2. Calculate u and b. Calculate $(u_1^t,\ b_1^t)$ by using (12) and $(u_2^t,\ b_2^t)$ by using (15) with $v_1 = v_1^t$ and $v_2 = v_2^t$.

3. Update v and d. Calculate $(v_1^{t+1},\ d_1^{t+1})$ by using (14) and $(v_2^{t+1},\ d_2^{t+1})$ by using (16) with $u_1 = u_1^t$ and $u_2 = u_2^t$.

4. Update u and b. Calculate $(u_1^{t+1},\ b_1^{t+1})$ by using (12) and $(u_2^{t+1},\ b_2^{t+1})$ by using (15) with $v_1 = v_1^{t+1}$ and $v_2 = v_2^{t+1}$.

5. If $\|u_1^{t+1} - u_1^t\|, \|u_2^{t+1} - u_2^t\|, \|v_1^{t+1} - v_1^t\|$ and $\|v_2^{t+1} - v_2^t\|$ are all less than ε or the maximum number of iterations is achieved, put $u_1^* \leftarrow u_1^{t+1}$, $v_1^* \leftarrow v_1^{t+1}$, $u_2^* \leftarrow u_2^{t+1}$, $v_2^* \leftarrow v_2^{t+1}$, $b_1^* \leftarrow \frac{b_1^{t+1} + d_1^{t+1}}{2}$ and $b_2^* \leftarrow \frac{b_2^{t+1} + d_2^{t+1}}{2}$; otherwise, set $t \leftarrow t + 1$ and return to Step 2.

6. Construct decision functions $f_i(X)$, $i = 1, 2$ by $f_1(X) = (u_1^*)^T \widetilde{X} v_1^* + b_1^*$ and $f_2(X) = (u_2^*)^T \widetilde{X} v_2^* + b_2^*$.

7. For a new input $\widetilde{X}$, its label $y_{\widetilde{X}}$ can be obtained by $y_{\widetilde{X}} = \arg \min_{i=1,2} \frac{|f_i(\widetilde{X})|}{\|u_i^*(v_i^*)^T\|}$.

3.2. NILS-TSMM

We know that up to now, nonlinear classifiers for matrix data are few. In order to solve this problem, in this subsection, we try to discuss the nonlinear version of ILS-TSMM by introducing a new matrix kernel function. Kernel skill is used to preprocess matrix data. Let $k : R^{n_1} \times R^{n_1} \to R$ be a kernel function, H (for the sake of consistency, denoted as R^∞) the reproducing kernel Hilbert space (RKHS) of the kernel k and $\varphi : R^{n_1} \to R^\infty$ the corresponding feature mapping. Let $\aleph = \{X_1, \cdots, X_m\}$, $A = \{X_1^1, \cdots, X_{m_1}^1\}$ and $B = \{X_1^2, \cdots, X_{m_2}^2\}$, where X_i^1 and X_j^2 express the i-th input belonging to positive class and the j-th input belonging to negative class, respectively.

We first introduce a matrix kernel function. For any matrix $X \in R^{n_1 \times n_2}$, it can be divided by columns as $X = [x_1, \cdots, x_{n_2}]$, where $x_i \in R^{n_1}$ is the i-th column of X. Put $\varphi(X) = [\varphi(x_1), \cdots, \varphi(x_{n_2})] \in R^{\infty \times n_2}$. For any given nonzero vector $v \in R^{n_2}$, we define a matrix kernel function $k_v : R^{n_1 \times n_2} \times R^{n_1 \times n_2} \to R$ with respect to v by

$$k_v(X, Z) =< \varphi(X)v, \varphi(Z)v >= v^T \varphi(X)^T \varphi(Z)v = v^T K_{XZ} v, \ \forall X, Z \in R^{n_1 \times n_2},$$

where $K_{XZ} = \varphi(X)^T \varphi(Z) = [\varphi(x_i)^T \varphi(z_j)]_{n_2 \times n_2} = [k(x_i, z_j)]_{n_2 \times n_2}$. To facilitate the following derivation, we set

$$K_v(A, \aleph) = \begin{bmatrix} k_v(X_1^1, X_1) & \cdots & k_v(X_1^1, X_m) \\ \vdots & \ddots & \vdots \\ k_v(X_{m_1}^1, X_1) & \cdots & k_v(X_{m_1}^1, X_m) \end{bmatrix} \in R^{m_1 \times m},$$

$$K_v(B, \aleph) = \begin{bmatrix} k_v(X_1^2, X_1) & \cdots & k_v(X_1^2, X_m) \\ \vdots & \ddots & \vdots \\ k_v(X_{m_2}^2, X_m) & \cdots & k_v(X_{m_2}^2, X_m) \end{bmatrix} \in R^{m_2 \times m},$$

$$K_v(\aleph, \aleph) = \begin{bmatrix} k_v(X_1, X_1) & \cdots & k_v(X_1, X_m) \\ \vdots & \ddots & \vdots \\ k_v(X_m, X_1) & \cdots & k_v(X_m, X_m) \end{bmatrix} \in R^{m \times m},$$

$$K_v(X, \aleph) = [k_v(X, X_1), \cdots, k_v(X, X_m)] \in R^{1 \times m}, \quad \forall X \in R^{n_1 \times n_2}.$$

For $k = 1, 2,$ we consider u_k in the subspace $span\{\varphi(X_1)v_k, \cdots, \varphi(X_m)v_k\}$ of R^∞ and assume that $u_k = \varphi_{v_k}(\aleph)\beta_k,$ where $\varphi_{v_k}(\aleph) = [\varphi(X_1)v_k, \cdots, \varphi(X_m)v_k] \in R^{\infty \times m}$ and $\beta_k \in R^m$. NILS-TSMM aims to seek a pair of nonparallel hyper-planes $u_1^T\varphi(X)v_1 + b_1 = 0$ and $u_2^T\varphi(X)v_2 + b_2 = 0$ by considering the following two QPPs:

$$\min_{u_1, v_1, b_1, \xi} \frac{c_3}{2}(\|u_1 v_1^T\|^2 + b_1^2) + \frac{1}{2}\sum_{i \in I_1}(u_1^T\varphi(X_i)v_1 + b_1)^2 + \frac{c_1}{2}\sum_{j \in I_2}\xi_j^2$$

$$s.t. \; -(u_1^T\varphi(X_j)v_1 + b_1) + \xi_j = 1, \quad j \in I_2, \tag{17}$$

$$\min_{u_2, v_2, b_2, \eta} \frac{c_4}{2}(\|u_2 v_2^T\|^2 + b_2^2) + \frac{1}{2}\sum_{j \in I_2}(u_2^T\varphi(X_j)v_2 + b_2)^2 + \frac{c_2}{2}\sum_{i \in I_1}\eta_i^2$$

$$s.t. \; (u_2^T\varphi(X_i)v_2 + b_2) + \eta_i = 1, \quad i \in I_1, \tag{18}$$

where $u_1, u_2 \in R^\infty$, $v_1, v_2 \in R^{n_2}$, and $b_1, b_2 \in R$.

Next, we utilize an alternating projection method to solve (u_i, v_i, b_i), $i = 1, 2$. For any given nonzero vector $v_1 \in R^{n_2}$, let $r_1 = \|v_1\|^2$, $\varphi_{v_1}(A) = [\varphi(X_1^1)v_1, \cdots, \varphi(X_{m_1}^1)v_1]$, $\varphi_{v_1}(B) = [\varphi(X_1^2)v_1, \cdots, \varphi(X_{m_2}^2)v_1]$, and $\varphi_{v_1}(\aleph) = [\varphi(X_1)v_1, \cdots, \varphi(X_m)v_1]$, then the problem (17) can be translated as

$$\min_{\beta_1, b_1, \xi} \frac{c_3}{2}(r_1\beta_1^T K_{v_1}(\aleph, \aleph)\beta_1 + b_1^2) + \frac{1}{2}\|K_{v_1}(A, \aleph)\beta_1 + b_1 e_1\|^2 + \frac{c_1}{2}\xi^T\xi$$

$$s.t. -(K_{v_1}(B, \aleph) + b_1 e_2) + \xi = e_2, \tag{19}$$

where $\|u_1\|^2 = \beta_1^T K_{v_1}(\aleph, \aleph)\beta_1$, $K_{v_1}(A, \aleph)\beta_1 = \varphi_{v_1}^T(A)u_1$ and $K_{v_1}(B, \aleph)\beta_1 = \varphi_{v_1}^T(B)u_1$. By solving the problem (19), we can get that

$$\begin{bmatrix} \beta_1 \\ b_1 \end{bmatrix} = -\left(\frac{c_3}{c_1} S_{v_1} + \frac{1}{c_1} P_1^T P_1 + Q_1^T Q_1\right)^{-1} Q_1^T e_2, \tag{20}$$

where $S_{v_1} = \begin{bmatrix} r_1 K_{v_1}(\aleph, \aleph), 0 \\ 0, \quad 1 \end{bmatrix}$, $P_1 = [K_{v_1}(A, \aleph), e_1]$ and $Q_1 = [K_{v_1}(B, \aleph), e_2]$. Once β_1 is calculated, we can get $u_1 = \varphi_{v_1}(\aleph)\beta_1$. Proceeding to the next step, we let

$$r_2 = \|u_1\|^2,$$

$$\varphi_{u_1}(A) = [\varphi^T(X_1^1)u_1, \cdots, \varphi^T(X_{m_1}^1)u_1],$$

$$\varphi_{u_1}(B) = [\varphi^T(X_1^2)u_1, \cdots, \varphi^T(X_{m_2}^2)u_1],$$

$$\varphi_{u_1}(\aleph) = [\varphi^T(X_1)u_1, \cdots, \varphi^T(X_m)u_1].$$

In this case, the problem (17) can be simplified as

$$\min_{v_1, d_1, \xi} \frac{c_3}{2}(r_2\|v_1\|^2 + d_1^2) + \frac{1}{2}\|\varphi_{u_1}^T(A)v_1 + e_1 d_1\|^2 + \frac{c_1}{2}\xi^T\xi$$

$$s.t. -(\varphi_{u_1}^T(B)v_1 + e_2 d_1) + \xi = e_2,$$

and then it can be deduced that

$$\begin{bmatrix} r_2 v_1 \\ d_1 \end{bmatrix} = -(\frac{c_3}{c_1} I + \frac{1}{c_1} P_2^T \widetilde{P_2} + Q_2^T \widetilde{Q_2})^{-1} Q_2^T e_2, \tag{21}$$

where $P_2 = [\varphi_{u_1}^T(A), e_1]$, $\widetilde{P_2} = [\frac{1}{r_2}\varphi_{u_1}^T(A), e_1]$, $Q_2 = [\varphi_{u_1}^T(B), e_2]$, and $\widetilde{Q_2} = [\frac{1}{r_2}\varphi_{u_1}^T(B), e_2]$.

By the similar way, we let $u_2 = \varphi_{v_2}(\aleph)\beta_2$ and can get from the problem (18) that

$$\begin{bmatrix} \beta_2 \\ b_2 \end{bmatrix} = (\frac{c_4}{c_2} S_{v_1} + \frac{1}{c_2} G_1^T G_1 + H_1^T H_1)^{-1} H_1^T e_1, \tag{22}$$

where $\alpha_1 = \|v_1\|^2$, $S_{v_2} = \begin{bmatrix} \alpha_1 K_{v_2}(\aleph, \aleph), & 0 \\ 0, & 1 \end{bmatrix}$, $G_1 = [K_{v_2}(B, \aleph), e_2]$, and $H_1 = [K_{v_2}(A, \aleph), e_1]$, and

$$\begin{bmatrix} \alpha_2 v_2 \\ d_2 \end{bmatrix} = (\frac{c_4}{c_2} I + \frac{1}{c_2} G_2^T \widetilde{G_2} + H_2^T \widetilde{H_2})^{-1} H_2^T e_1, \tag{23}$$

where $\alpha_2 = \|u_2\|^2, G_2 = [\varphi_{u_2}^T(B), e_2], \widetilde{G_2} = [\frac{1}{\alpha_2}\varphi_{u_2}^T(B), e_2], H_2 = [\varphi_{u_2}^T(A), e_1]$, and $\widetilde{H_2} = [\frac{1}{\alpha_2}\varphi_{u_2}^T(A), e_1]$.

Consequently, (β_1, v_1) and (β_2, v_2) can be obtained by iteratively solving (20)-(21) and (22)-(23), respectively. The specific procedure is as follows.

Algorithm 3: NILS-TSMM

1. Initialization: Let $t = 0$, $\varepsilon > 0$ small enough and take $v_1^t = v_2^t = [1, \cdots, 1]^T$.

2. Calculate β and b. Calculate (β_1^t, b_1^t) by using (20) and (β_2^t, b_2^t) by using (22) with $v_1 = v_1^t$ and $v_2 = v_2^t$.

3. Update v and d. Calculate (v_1^{t+1}, d_1^{t+1}) by using (21) and (v_2^{t+1}, d_2^{t+1}) by using (23) with $u_1 = u_1^t$ and $u_2 = u_2^t$.

4. Update β and b. Calculate $(\beta_1^{t+1}, b_1^{t+1})$ by using (20) and $(\beta_2^{t+1}, b_2^{t+1})$ by using (22) with $v_1 = v_1^{t+1}$ and $v_2 = v_2^{t+1}$.

5. If $\|\beta_1^{t+1} - \beta_1^t\|, \|\beta_2^{t+1} - \beta_2^t\|, \|v_1^{t+1} - v_1^t\|$ and $\|v_2^{t+1} - v_2^t\|$ are all less than ε or the maximum number of iterations is achieved, put $\beta_1^* \leftarrow \beta_1^{t+1}$, $v_1^* \leftarrow v_1^{t+1}$, $\beta_2^* \leftarrow \beta_2^{t+1}$, $v_2^* \leftarrow v_2^{t+1}$, $b_1^* \leftarrow \dfrac{b_1^{t+1} + d_1^{t+1}}{2}$, and $b_2^* \leftarrow \dfrac{b_2^{t+1} + d_2^{t+1}}{2}$; otherwise, set $t \leftarrow t + 1$ and return to Step 2.

6. Calculate $\|u_i^*(v_i^*)^T\|^2$ by $\|u_i^*(v_i^*)^T\|^2 = (\beta_i^*)^T K_{v_i^*}(\aleph, \aleph)\beta_i^* \|v_i^*\|^2$, $i = 1, 2$.

7. Construct decision functions $f_i(X)$, $i = 1, 2$ by $f_i(X) = [k_{v_i^*}(X_1, X), \cdots, k_{v_i^*}(X_m, X)]\beta_i^* + b_i^*$.

8. For a new input $\widetilde{X}$, its label $y_{\widetilde{X}}$ can be obtained by $y_{\widetilde{X}} = \arg \min\limits_{i=1,2} \dfrac{|f_i(\widetilde{X})|}{\|u_i^*(v_i^*)^T\|}$.

4. Experiments and Results Analysis

In this section, in order to demonstrate the effectiveness of ILS-TSMM and NILS-TSMM, we perform comparative experiments with linear LS-TSVM and linear STM on both classification percent accuracy and computation time (including training time and testing time) on ORL and Yale face databases [32-33]. Since nonlinear classifiers for matrix data are few, we can only compare ILS-TSMM and NILS-TSMM for illustrating the superiority of NILS-TSMM.

ORL face database contains 400 face images of 40 individuals taken between April 1992 and April 1994 at different times, light and facial expressions. Each individual has 10 face images. Yale face database contains 165 face images of 15 individuals with 11 images for each one. For the convenience of calculation, we chose 13 individuals from ORL database and 7 individuals from Yale database according to different facial details for our experiments. Related information of the selected images is described in Tables 1 and 2. We use 10-fold cross validation method in the experiments. All the experiments are implemented in MATLAB (R2012b) running on a PC with system configuration Intel(R) Core(TM) i3 (2.53GHz) with 2GB of RAM.

4.1. Experiments on ORL database

In this subsection, we perform the comparative experiments on 8 pairs composed from 13 individuals taken from ORL database. In each pair, we select 1, 5, and 9 images for each individual as training samples and the others as testing samples. In all experiments, take $\varepsilon = 10^{-3}$, $c_1 = c_2 = 0.15$, $c_3 = c_4 = 0.1$ and each image is cropped into 28×23 pixels. In NILS-TSMM, Gaussian RBF kernel $k(x, y) = \exp\{ - \frac{\|x - y\|^2}{\sigma^2} \}$ is used and grid search approach from 10^3 to 10^5 is employed for selecting the optimal kernel parameter σ. Experiment results are shown in Tables 3-4, in which acc, time and train represent percent accuracy,

running time (second) and the number of training samples, respectively. The comparison results of three linear classifiers are showed in Table 3 and the comparison results between ILS-TSMM and NILS-TSMM are showed in Table 4.

From Table 3, we can see that the computation time of ILS-TSMM is obviously less than that of LS-TSVM and STM. For 5 or 9 training samples case, the computation time of ILS-TSMM is almost faster 2-3 times than LS-TSVM and 4-8 times than STM, and the classification accuracy of ILS-TSMM is better than that of LS-TSVM and almost the same with that of STM. For one training sample case, the classification accuracies of three classifiers are the competition each other. From Table 4, we can see that the classification accuracy of NILS-TSMM is higher than that of three linear classifiers in general. Especially, the classification accuracy of NILS-TSMM is 100% in the case of 9 training samples. In addition, along with the increase of the number of training samples, the classification accuracy of ILS-TSMM and MILS-TSMM is increased obviously.

Table 1. 13 individuals chosen from ORL database

Facial detail	1	5	7	8	13	16	17	22	24	25	29	34	37
pic													
Bald	no	no	no	no	yes	no	no	no	no	no	no	yes	yes
Mustache	no	no	light	no	no	light	light	no	no	light	no	no	heavy
FaceSize	thin	mid	mid	mid	thin	thin	mid	fat	thin	mid	mid	fat	mid
Age	young	young	young	young	mid	young	mid	mid	mid	old	young	old	mid
Glasses	yes	no	yes	no	yes	no	yes	no	no	no	no	yes	yes

Table 2. 7 individuals chosen from Yale database

Facial detail	2	4	7	8	9	11	13
pic							
Mustache	light	no	heavy	no	light	no	light
FaceSize	thin	mid	mid	mid	mid	fat	thin
Age	mid	young	mid	young	mid	mid	young
Glasses	no	no	no	yes	no	no	yes

Table 3. Accuracy and computing time of three linear classifiers on ORL database

Pairs	Classifiers	acc(%)-time(s)	acc(%)-time(s)	acc(%)-time(s)
		Train = 1	Train = 5	Train = 9
(1,5)	ILS-TSMM	88.89 ± 1.17 **(0.66)**	**100.00 ± 0.00 (1.27)**	**100.00 ± 0.00 (3.45)**
	LS-TSVM	**90.56 ± 0.62**(8.60)	100.00 ± 0.00(9.38)	100.00 ± 0.00(9.48)
	STM	86.67 ± 1.25(5.06)	99.00 ± 0.10(26.06)	100.00 ± 0.00(42.17)
(7,17)	ILS-TSMM	**87.22 ± 1.86 (0.82)**	**100.00 ± 0.00 (2.86)**	**100.00 ± 0.00 (3.44)**
	LS-TSVM	87.22 ± 1.51(10.32)	100.00 ± 0.00(6.89)	100.00 ± 0.00(12.67)
	STM	80.00 ± 2.07(11.94)	77.00 ± 6.01(26.47)	100.00 ± 0.00(40.09)
(8,24)	ILS-TSMM	**98.33 ± 0.14(0.99)**	**100.00 ± 0.00 (1.79)**	**100.00 ± 0.00 (3.40)**
	LS-TSVM	97.78 ± 0.08(9.23)	100.00 ± 0.00(14.26)	100.00 ± 0.00(13.00)
	STM	98.33 ± 0.07(3.94)	100.00 ± 0.00(26.76)	100.00 ± 0.00(42.00)
(13,16)	ILS-TSMM	92.22 ± 0.29 **(0.34)**	**100.00 ± 0.00 (0.96)**	**100.00 ± 0.00 (3.31)**
	LS-TSVM	**93.33 ± 0.26**(9.99)	100.00 ± 0.00(17.45)	100.00 ± 0.00(16.47)
	STM	84.44 ± 1.15(3.87)	89.00 ± 3.43(25.68)	100.00 ± 0.00(41.22)
(13,22)	ILS-TSMM	**100.00 ± 0.00(0.46)**	**100.00 ± 0.00 (0.48)**	**100.00 ± 0.00 (3.37)**
	LS-TSVM	99.44 ± 0.03(9.59)	100.00 ± 0.00(9.24)	100.00 ± 0.00(12.25)
	STM	100.00 ± 0.00(1.88)	100.00 ± 0.00(31.04)	100.00 ± 0.00(40.10)
(13,34)	ILS-TSMM	**100.00 ± 0.00(0.35)**	**100.00 ± 0.00 (4.65)**	**100.00 ± 0.00 (3.42)**
	LS-TSVM	100.00 ± 0.00(8.32)	100.00 ± 0.00(6.80)	100.00 ± 0.00(16.06)
	STM	100.00 ± 0.00(3.55)	100.00 ± 0.00(26.19)	100.00 ± 0.00(41.49)
(13,37)	ILS-TSMM	**93.89 ± 0.51(0.56)**	**100.00 ± 0.00 (1.15)**	**100.00 ± 0.00 (3.45)**
	LS-TSVM	78.33 ± 1.33(10.16)	100.00 ± 0.00(9.92)	100.00 ± 0.00(14.62)
	STM	86.67 ± 1.39(3.73)	100.00 ± 0.00(30.42)	100.00 ± 0.00(42.75)
(25,29)	ILS-TSMM	**100.00 ± 0.00(0.35)**	96.00 ± 0.71 **(1.11)**	**100.00 ± 0.00 (4.21)**
	LS-TSVM	99.44 ± 0.03(16.60)	**100.00 ± 0.00**(17.20)	100.00 ± 0.00(13.97)
	STM	100.00 ± 0.00(3.35)	100.00 ± 0.00(29.56)	100.00 ± 0.00(39.51)

Table 4. Accuracy of linear and nonlinear ILS-TSMM on ORL database

Pairs	Classifiers	acc(%)	acc(%)	acc(%)
		Train = 1	Train = 5	Train = 9
(1,5)	NILS-TSMM	**98.00 ± 0.18**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	88.89 ± 1.17	100.00 ± 0.00	100.00 ± 0.00
(7,17)	NILS-TSMM	**88.00 ± 1.34**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	87.22 ± 1.86	100.00 ± 0.00	100.00 ± 0.00
(8,24)	NILS-TSMM	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	98.33 ± 0.14	100.00 ± 0.00	100.00 ± 0.00
(13,16)	NILS-TSMM	**94.00 ± 0.27**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	92.22 ± 0.29	100.00 ± 0.00	100.00 ± 0.00
(13,22)	NILS-TSMM	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
(13,34)	NILS-TSMM	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
(13,37)	NILS-TSMM	**94.50 ± 0.47**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	93.89 ± 0.51	100.00 ± 0.00	100.00 ± 0.00
(25,29)	NILS-TSMM	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	100.00 ± 0.00	96.00 ± 0.71	100.00 ± 0.00

Table 5. Accuracy and computing time of three linear classifiers on Yale database

Pairs	Classifiers	acc(%)-time(s)	acc(%)-time(s)	acc(%)-time(s)
		Train = 1	Train = 5	Train = 9
	ILS-TSMM	**95.00 ± 2.50(1.09)**	**100.00 ± 0.00 (0.73)**	**100.00 ± 0.00 (1.75)**
(2,4)	LS-TSVM	95.00 ± 1.95(4.83)	100.00 ± 0.00(6.77)	100.00 ± 0.00(5.69)
	STM	92.50 ± 2.51(5.72)	91.67 ± 0.31(7.26)	97.50 ± 0.63(15.07)
	ILS-TSMM	86.67 ± 2.89 **(1.49)**	**100.00 ± 0.00 (1.40)**	**100.00 ± 0.00 (2.53)**
(2,7)	LS-TSVM	**91.67 ± 2.28** (4.32)	100.00 ± 0.00(11.99)	100.00 ± 0.00(5.27)
	STM	87.00 ± 2.46(3.45)	98.33 ± 0.12(7.68)	95.00 ± 2.50(10.23)
	ILS-TSMM	**100.00 ± 0.00(1.38)**	**100.00 ± 0.00 (2.50)**	**100.00 ± 0.00 (2.83)**
(2,11)	LS-TSVM	100.00 ± 0.00(5.41)	100.00 ± 0.00(3.83)	100.00 ± 0.00(7.86)
	STM	100.00 ± 0.00(4.11)	100.00 ± 0.00(3.98)	100.00 ± 0.00(4.65)
	ILS-TSMM	**92.22 ± 2.14(2.03)**	**100.00 ± 0.00 (2.62)**	**100.00 ± 0.00 (2.74)**
(7,13)	LS-TSVM	92.22 ± 1.52(4.04)	100.00 ± 0.00(3.71)	100.00 ± 0.00(3.55)
	STM	58.00 ± 2.57(5.84)	95.83 ± 0.19(4.32)	100.00 ± 0.00(6.94)
	ILS-TSMM	81.11 ± 3.24**(0.13)**	**100.00 ± 0.00 (1.01)**	**100.00 ± 0.00 (2.85)**
(8,13)	LS-TSVM	**91.11 ± 2.07**(6.61)	100.00 ± 0.00(8.60)	100.00 ± 0.00(13.07)
	STM	75.00 ± 3.06(5.69)	84.17 ± 1.77(6.72)	97.50 ± 0.63(10.13)
	ILS-TSMM	**99.44 ± 0.03(1.23)**	**100.00 ± 0.00 (0.54)**	**100.00 ± 0.00 (0.40)**
(9,13)	LS-TSVM	98.89 ± 0.05(4.98)	100.00 ± 0.00(11.40)	100.00 ± 0.00(5.08)
	STM	89.50 ± 1.03(3.47)	92.50±0.07(5.50)	100.00 ± 0.00(7.06)

Table 6. Accuracy of linear and nonlinear ILS-TSMM on Yale database

Pairs	Classifiers	Train = 1	Train = 5	Train = 9
(2,4)	NILS-TSMM	94.00 ± 0.88	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	**95.00 ± 2.50**	100.00 ± 0.00	100.00 ± 0.00
(2,7)	NILS-TSMM	**96.00 ± 0.16**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	86.67 ± 2.89	100.00 ± 0.00	100.00 ± 0.00
(2,11)	NILS-TSMM	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
(7,13)	NILS-TSMM	**93.50 ± 1.06**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	92.22 ± 2.14	100.00 ± 0.00	100.00 ± 0.00
(8,13)	NILS-TSMM	**87.50 ± 2.24**	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	81.11 ± 3.24	100.00 ± 0.00	100.00 ± 0.00
(9,13)	NILS-TSMM	96.00 ± 1.21	**100.00 ± 0.00**	**100.00 ± 0.00**
	ILS-TSMM	**99.44 ± 0.03**	100.00 ± 0.00	100.00 ± 0.00

4.2. Experiments on Yale database

In this subsection, we perform the comparative experiments on 6 pairs composed from 7 individuals taken from Yale database. In each pair, we select 1, 5 or 9 images for each individual as training samples and the others as testing samples. In all experiments, take $\varepsilon = 10^{-3}$, $c_1 = c_2 = 0.15$, $c_3 = c_4$ and each image is cropped into 25×20 pixels, for c_3 we use grid search approach from 2^{-8} to 2^8 to select the best parameter. In NILS-TSMM, Gaussian RBF kernel $k(x, y) = \exp\{-\frac{\|x - y\|^2}{\sigma^2}\}$ is used and grid search approach from 2^{-8} to 2^8 is employed for selecting the optimal kernel parameter σ. The comparison results of three linear classifiers are showed in Table 5 and the comparison results of ILS-TSMM and NILS-TSMM are showed in Table 6.

From Tables 5-6, we can see that the computation time of ILS-TSMM is apparently less than that of LS-TSVM and STM. For 5 or 9 training samples case, the classification accuracies of both ILS-TSMM and NILS-

TSMM are almost the same each other, and they are higher than that of LS-TSVM and STM. For one training sample case, the classification accuracies of three linear classifiers are the competition each other, and the classification accuracy of NILS-TSMM is higher than that of ILS-TSMM in general. In addition, along with the increase of the number of training samples, the classification accuracy of both ILS-TSMM and NILS-TSMM is increased obviously.

5. Conclusion

In this paper, we propose ILS-TSMM and NILS-TSMM for matrix data classification. In ILS-TSMM, we seek directly a pair of nonparallel hyper-planes by solving 4 equalities rather than solving two dual QPPs, which reduces greatly the computation time. It notes that up to now nonlinear classification methods for matrix data are few. In order to consider the nonlinear version of ILS-TSMM, we first introduce a new matrix kernel function and then, by means of the defined matrix kernel function, provide a detailed theoretical derivation for NILS-TSMM.

In order to check the effectiveness of both ILS-TSMM and NILS-TSMM, we perform comparative experiments with linear LS-TSVM and STM on 14 group binary classification problems taken from ORL and Yale face databases. From the experiment results in Tables 3-6, we can conclude that ILS-TSMM is more effective than LSTSVM and STM on both classification accuracy and computation time, and NILS-TSMM is more effective than three linear classifiers on classification accuracy in general. In addition, along with the increase of the number of training samples, the classification accuracies of both ILS-TSMM and NILS-TSMM are increased obviously. In this issue, there are lots of works to do, such as, generalization and improvement of algorithms, optimization selection of parameters in modellings and so on.

References

[1] V. N. Vapnik, Statistical Learning Theory, Springer, 1998.

[2] V. N. Vapnik, The Nature of Statistical Learning Theory, New York, NY, USA: Springer-Verlag, 1995.

[3] Jayadeva, R. Khemchandani and S. Chandra, Twin support vector machines for pattern classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 29(5) (2007), 905-910.

[4] Y. H. Shao, C. H. Zhang, X. B. Wang and N. Y. Deng, Improvements on twin support vector machines, IEEE Transactions on Neural Networks 22(6) (2011), 962-968.

[5] J. A. K. Suykens, L. Lukas and P. Van Dooren et al., Least squares support vector machine classifiers: A large scale algorithm, In Proceedings of European Conference Circuit Theory Design (1999), 839-842.

[6] J. A. K. Suykens and J. Vandewalle, Least squares support vector machine classifiers, Neural Processing Letters 9(3) (1999), 293-300.

[7] M. A. Kumar and M. Gopal, Least squares twin support vector machines for pattern classification, Expert Systems with Applications 36(4) (2009), 7535-7543.

[8] S. Balasundaram and Kapil, Application of lagrangian twin support vector machines for classification, Second International Conference on Machine Learning and Computing (2010), 193-197.

[9] Z. Q. Qi, Y. J. Tian and Y. Shi, Structural twin support vector machine for classification, Knowledge-Based Systems 43 (2013), 74-81.

[10] Pen Xinjun, Least squares twin support vector hypersphere (LS-TSVH) for pattern recognition, Expert Systems with Applications 37 (2010), 8371-8378.

[11] S. Ghorai, A. Mukherjee and P. K. Dutta, Nonparallel plane proximal classifier, Signal Processing 89 (2009), 510-522.

[12] M. A. Kumar and M. Gopal, Application of smoothing technique on twin support vector machines, Pattern Recognition Letters 29 (2008), 1842-1848.

[13] O. L. Mangasarian and E. W. Wild, Multisurface proximal support vector classification via generalized eigenvalues, IEEE Transactions on Pattern Analysis and Machine Intelligence 28(1) (2006), 69-74.

[14] E. Osuna, R. Freund and F. Girosi, Training support vector machines: An application to face detection, In Proceedings of IEEE Computer Vision and Pattern Recognition (1997), 130-136.

[15] X. Peng, Twin support vector hypersphere (TSVH) classifier for pattern recognition, Neural Networks.

[16] Z. Qi, Y. Tian and S. Yong, Robust twin support vector machine for pattern classification, Pattern Recognition 46(1) (2013), 305-316.

[17] Z. Qi, Y. Tian and S. Yong, Laplacian twin support vector machine for semisupervised classification, Neural Networks 35 (2012), 46-53.

[18] Y. T. Xu, An improved least squares twin support vector machine, Journal of Information & Computational Science 4 (2012), 1063-1071.

[19] J. Q. Gao, L. Y. Fan, L. Li and L. Z. Xu, A practical application of kernel-based fuzzy discriminant analysis, Int. J. Appl. Math. Comput. Sci. 23(4) (2013), 887-903.

[20] J. Q. Gao, L. Z. Xu, A. Shi and F. C. Huang, A kernel-based block matrix decomposition approach for the classification of remotely sensed images, Applied Mathematics and Computation 228 (2014), 531-545.

[21] D. Tao, X. Li and X. Wu et al., Supervised tensor learning, Knowledge and Information Systems 13(1) (2007), 1-42.

[22] D. Cai, X. He, J. R. Wen, J. Han and W. Y. Ma, Support tensor machines for text categorization, Department of Computer Science, UIUC. UIUCDCS-R 2006-2714.

[23] C. Hou et al., Multiple rank multi-linear SVM for matrix data classification, Pattern Recognition 47 (2014), 454-469.

[24] R. Khemchandani, A. Karpatne and S. Chandra, Proximal support tensor machines, International Journal of Machine Learning and Cybernetics 4(6) (2013), 703-712.

[25] Xinsheng Zhang, Xinbo Gao and Ying Wang, Twin support tensor machines for MCs detection, Journal of Electronics (China) 26(3) (2009), 318-325.

[26] G. Fung and O. L. Mangasarian, Proximal support vector machine classifiers, In Proceedings of Seventh International Conference on Knowledge and Data Discovery (2001), 77-86.

[27] D. Tao, X. Li, W. Hu, S. J. Maybank and X. Wu, General tensor discriminant analysis and Gabor features for gait recognition, IEEE Trans. Pattern Anal. Mach. Intell. 29(10) (2007), 1700-1715.

[28] D. Tao, X. Li, W. Hu, S. J. Maybank and X. Wu, Tensor rank one discriminant analysis: A convergent method for discriminative multilinear subspace selection, Neurocomputing 71(10-12) (2008), 1866-1882.

[29] D. Tao, M. Song, X. Li, J. Shen, J. Sun, X. Wu, C. Faloutsos and S. J. Maybank, Bayesian tensor approach for 3-D face modelling, IEEE Trans. Circuits. Syst. Video Technol. 18(10) (2008), 1397-1410.

[30] D. Tao, J. Sun, J. Shen, X. Wu, X. Li, S. J. Maybank and C. Faloutsos, Bayesion tensor analysis, IEEE International Joint Conference on Date of Conference Neural Networks (1-8) (2008), 1402-1409.

[31] J. Sun, D. Tao, S. Papadimitriou, P. Yu and C. Faloutsos, Incremental tensor analysis: Theory and applications, ACM Transactions on Knowledge Discovery from Data TKDD 2(3) (2008).

[32] AT&T Labs Cambridge, The Olivetti and Oracle Research Laboratory Database of Faces, 1994.

http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[33] http://cvc.yale.edu/projects/yalefaces/yalefaces.html.

■